Los Alamos
NATIONAL LABORATORY
research note
Computer and Computational Sciences
CCS–4:Transport Methods Group

To/MS: Distribution
From/MS: Todd Urbatsch, CCS–4, MS D409;
Tom Evans, CCS–4, MS D409;
Mike Buksas, CCS–4, MS D409
Phone/FAX: (505)667–3513, 665–3677, 667–7580
Symbol: CCS–4:02-13(U) (LA-UR-02-3213)
Date: May 28, 2002

Subject: Release Notification: MILAGRO–3_0_0: Multigroup

Executive Summary

We wish to announce the release of version 3_0_0 of Milagro, the parallel, multi-dimensional, multi-geometry, stand-alone Implicit Monte Carlo code for thermal radiative transfer. This release of Milagro has a multigroup frequency treatment. The multigroup opacities are either input as analytic models or read from IPCRESS opacity files. In this note we briefly describe the capability, assumptions, algorithms, software design, input details, and the results from verification test problems.

1. Introduction

We have introduced a multigroup frequency capability in the Milagro Implicit Monte Carlo (IMC) code [1–6]. This capability represents additional functionality beyond the gray frequency treatment in all prior versions.

We describe the assumptions built into Milagro's multigroup frequency treatment, the algorithms used for frequency integration, and the schemes for sampling a particle's frequency group.

We also describe the new C++ techniques we devised. These techniques allowed us to retain the gray capability alongside the new multigroup capability without sacrificing speed or safety and without adding redundant coding.

The additional input necessary to run the multigroup Milagro executable is presented. In particular, we explain how to run Milagro with user-input analytic opacities or with data (real or analytic) from an IPCRESS [7] opacity file.

The multigroup capability has been tested and verified to be correct with a myriad of component tests, degenerate consistency tests, and analytic benchmarks.

2. Assumptions

In our multigroup frequency treatment, we have made certain assumptions about frequency distributions, the spatial dependence of the frequency distributions, and the range of frequency integrations.

2.1. **Sampled Spectrum of Emission Particles**

In the Fleck and Cummings IMC method [8], the time-explicit portion of the volume emission source is emitted according to the temperature of the cell. A flat spatial dependence of the temperature within a cell produces unacceptable spatial inaccuracies, thus necessitating a first- or higher-order treatment of the spatial dependence of temperature within a cell [9]. In Milagro, the position of a volume emission source is sampled according to a linear function of T^4 within each cell. Therefore, more particles are sampled from the hotter side of the cell.

Milagro's multigroup treatment does not apply the in-cell spatial dependence of temperature to the frequency sampling. The frequency of each volume emission particle in a cell is sampled according to the average temperature of the cell. So, the particles sampled from the hot side of cell have the same frequency distribution as those sampled from the cold side of a cell.

2.2. **Assumed Planckian Distributions**

Milagro assumes a Planckian spectral shape for its sources. In Milagro's multigroup frequency treatment, particles due to volume emission and the time-implicit portion of an external material source have frequencies that are sampled from a discrete, opacity–weighted Planckian distribution [8]. The surface source is assumed to come from a blackbody at T_{ss}, so, naturally and rigorously, surface source particles are sampled from a straight Planckian at that temperature. The frequencies of the initial census particles are sampled from a Planckian at the initial radiation temperature in each cell. (Initial census particles are sampled uniformly in space.)

Finally, since there is no temperature associated with a radiation source, the frequency of particles due to an external radiation source is sampled from a Planckian distribution evaluated at the *material* temperature. Thus, the radiation source has built-in assumptions of local thermodynamic equilibrium (LTE). Future Milagro improvements may include a user-specified frequency dependence of the radiation source.

2.3. **Frequency Limits of Planckian Integrals**

The calculations of source energies involve integrals over the range of frequencies considered. In a gray treatment, the normalized Planckian function is already integrated from 0 to ∞ to produce a value of unity. In a multigroup frequency treatment, the total integral of the normalized Planckian extends over a range less than $(0, \infty)$ and will be less than unity. Poorly chosen upper and lower frequency bounds could have the following consequences:

- the spectrum and corresponding energy in parts of phase space will go unsampled,
- inability to sample a frequency when the Planckian functions are outside the upper or lower frequency, e.g., a cold cell when the lower group bound is too high,
- the frequency limits may not allow a consistent conversion between energy and temperature. For example, the energy rates input for external sources may have been ascertained with frequency limits different than what the calculation is actually using.

We also note that whenever a radiation temperature is output from Milagro, it is a quantity derived from the radiation energy density assuming a Planckian shape integrated over all frequencies from 0 to ∞, i.e., $E_r = aT_r^4$.

3. Algorithms

3.1. Brad Clark's Finite Planckian Integrals

We have incorporated static functions in the Common Data Interface (CDI) package [7, 10] to integrate the normalized Planckian function between two frequencies. These functions use the algorithms developed by Bradley A. Clark [11,12]. Clark's algorithm can have any desired accuracy. It is an improvement over Zimmerman's rational polynomial fit [11] that is used by earlier IMC efforts at LANL.

Calculating IMC source energies requires integrals of the Planckian over the entire frequency range. Planckian integrals are also required for each individual frequency group for the emission cumulative distribution function (cdf), which is an opacity–weighted Planckian.

3.2. Multigroup Source Energy Calculations

In a gray calculation, the equations are integrated over the frequency range $(0, \infty)$. Given that the sources in Milagro have an assumed Planckian spectrum, the source energies in a gray frequency treatment contain the integral of the Planckian over all frequencies from 0 to ∞, the value of which is acT^4. The common practice is to represent the Planckian in terms of the normalized Planckian, $b_\nu(T)$,

$$B_\nu(T) = acT^4 \, b_\nu(T) \quad , \tag{1}$$

where $b_\nu(T)$ is non-negative and the integral of $b_\nu(T)$ from $\nu = 0$ to ∞ is unity. For G groups, a minimum group boundary of $\nu_0 > 0$, and a maximum group boundary of $\nu_G < \infty$, the full integral of $b_\nu(T)$ is less than one.

Thus, we find that the multigroup source energies are merely scaled values of the gray source energies. For purely Planckian sources, such as a surface source, the multigroup source energy contains the following integral:

$$\int_{\nu_0}^{\nu_G} B_\nu(T) \, d\nu = acT^4 \int_{\nu_0}^{\nu_G} b_\nu(T) \, d\nu \quad , \tag{2}$$

where the integral of the normalized Planckian is evaluated using Clark's algorithm.

For an emission or re-emission source, the energy is proportional to the integral of the opacity–weighted Planckian distribution. Given a piece-wise constant multigroup opacity, σ_g, where $g \in$

$[1, G]$, we evaluate the integral in the following manner:

$$\int_{\nu_0}^{\nu_G} \sigma B_\nu(T)\, d\nu = \sum_{g=1}^{G} \int_{\nu_{g-1}}^{\nu_g} \sigma B_\nu(T)\, d\nu \tag{3}$$

$$= \sum_{g=1}^{G} \sigma_g \int_{\nu_{g-1}}^{\nu_g} B_\nu(T)\, d\nu \tag{4}$$

$$= acT^4 \sum_{g=1}^{G} \sigma_g b_g \ , \tag{5}$$

where

$$b_g = \int_{\nu_{g-1}}^{\nu_g} b_\nu(T)\, d\nu \ . \tag{6}$$

Again, Clark's algorithm is used to evaluate b_g.

3.3. Sampling a Frequency Group

At this time, Milagro is purely multigroup since its frequency variable is truly discrete. The particles only have a group number, not an explicit frequency. With future introductions of such physics as Compton scattering and a material motion treatment, Milagro will require a within-group frequency treatment such that each particle will have both an explicit frequency and group.

3.3.1. *Sampling a Group from a Planckian.* For sampling a frequency group from a pure Planckian distribution, we chose an algorithm by taking into consideration Milagro's future requirements of continuous within-group frequencies. First, we sample a frequency from a continuous Planckian distribution, then we determine the group in which the sampled frequency resides. (An alternative for sampling a group only is to construct and sample from a discrete cumulative distribution function (cdf) equal to the running sum of b_g for $g = 1, ..., G$.)

To sample a frequency from the continuous Planck distribution, we use Barnett and Canfield's truncated infinite series technique [13] . They begin by writing the normalized Planckian, $b(x) = (15/\pi^4)x^3/(e^x - 1)$, as $\sum_{n=1}^{\infty} p_n f_n(x)$, where $x = h\nu/(kT)$ is the reduced frequency, $p_n = 90/(\pi^4 n^4)$, $f_n(x) = (n^4/6)x^3 e^{-nx}$, and $\sum_{n=1}^{\infty} p_n = 1$. Thus, with probability p_n we can sample $f_n(x)$ for x. Sampling n from p_n requires satisfying $(\pi^4/90)\xi \le \sum_{k=1}^{n} 1/k^4$, where ξ is a random number between 0 and 1. The search usually only requires one iteration (i.e., $n = 1$), but the search can be unbounded. The search is bounded by truncating (rounding down) the decimal representation of $\pi^4/90 = 1.0823232337....$ The maximum number of iterations for a given truncation is shown in Table 1. A large number of iterations, n, usually corresponds to a small frequency, so the concern over significant digits is largely moot. However, given that the maximum number of iterations is rarely encountered, the computational cost is not excessive (but the expenditure of random numbers might get wasteful). Milagro currently has 8 significant digits in its value for $\pi^4/90$.

Once n is found, four random numbers are used to sample x from $f_n(x)$ according to the procedure in Everett and Cashwell's "A Third Monte Carlo Sampler" [14]:

$$x = -\frac{1}{n} \log(\xi_1 \xi_2 \xi_3 \xi_4) \ . \tag{7}$$

TABLE 1: Maximum number of iterations for sampling a frequency from a continuous Planckian distribution for a given number of significant digits in $\pi^4/90$.

Number of Significant Digits	Truncated $\pi^4/90$	Max iterations
6	1.08232	47
7	1.082323	113
8	1.0823232	215
9	1.08232323	448
10	1.082323233	775
11	1.0823232337	2744

The frequency, then, is $h\nu = x\ kT$.

The sampling procedure was verified with a python script (script, data, and plots are located at /home/tmonster/methods/imc/multigroup). Figure 1 shows the sampled Planckian for a temperature of 1 keV and 10^6 particles. Figure 2 shows the relative error in the binned samples.

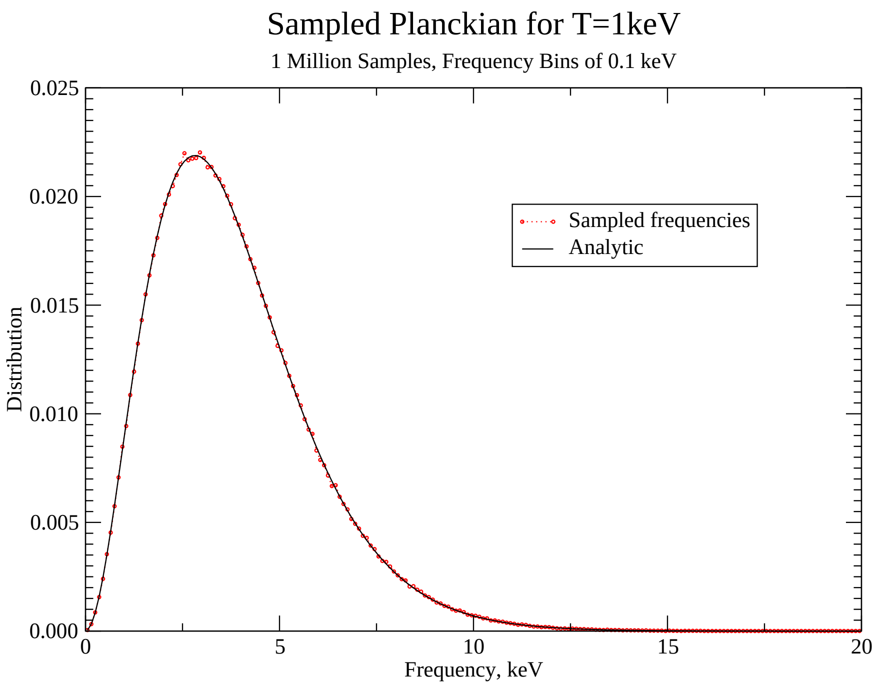

FIG. 1: Binned distribution of frequencies sampled from a continuous Planckian distribution at 1 keV.

The group containing the sampled frequency is determined by searching the group boundaries with a binary search. If the frequency is outside the multigroup boundaries, the frequency is rejected and resampled. If 100 rejections occur, the code fires an assertion and aborts. Rejections will occur when, for example, the minimum group bound is too high to capture the Planckian of a cold temperature.

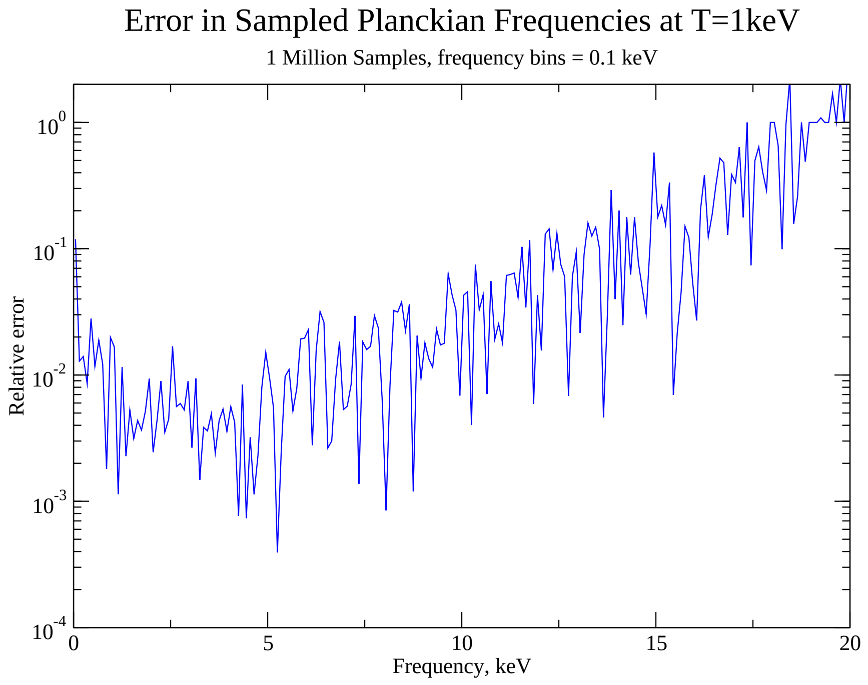

FIG. 2: Relative error of sampled and binned Planckian frequencies.

3.3.2. *Sampling a Group from an Emission Spectrum.* For emission spectra, a particle's frequency group must be sampled from the piece-wise constant distribution of $\sigma_g b_g$. We store the unnormalized cumulative distribution function (cdf),

$$f_i = \sum_{1}^{i} \sigma_g b_g \tag{8}$$

for each cell's temperature. The cdf is calculated with piece-wise constant opacities and the group-wise normalized Planckian integrals from Brad Clark's algorithm.

At this time, we do not further sample a continuous frequency within the sampled group. When we are required to do so, we may choose from one of many schemes:

- select $h\nu$ at the midpoint of the group,
- select $h\nu$ at the logarithmic midpoint of the group,
- sample $h\nu$ uniformly in the group,
- sample $h\nu$ linearly in the group,
- sample $h\nu$ from x^3 in low groups; uniformly in middle groups; and from an exponential in higher groups,
- if possible, sample $h\nu$ analytically from a Planckian distribution within the group using Clark's expansions.

3.4. **Differences Between Running Multigroup and Gray IMC**

The user should be aware of a few differences between multigroup and gray IMC. The multigroup parts of IMC entail a little extra work. The work is such that parallel behavior may be different than that of gray IMC. Also, the extra frequency dimension affects statistical noise.

3.4.1. *Extra Multigroup Work.*

The multigroup capability in the Milagro IMC code requires extra work in three areas:

- calculating the emission cdf (opacity-weighted Planckian for each group and each cell)
- sampling a group from a pure Planckian function (surface source, radiation source, and initial census particles)
- sampling a group from an emission cdf (volume emission and material source particles, and Fleck and Cummings' effective scatter)

In most cases, the work required to sample a group will be negligible compared to the rest of the transport work. Even if an IMC calculation is dominated by effective scatter, the extra sampling time is negligible because sampling a cdf is very fast (throw a random number, and perform a binary search for the group).

The calculation of the emission cumulative distribution function (cdf) is performed for each group and in each cell. This calculation is part of the deterministic source calculation required to run IMC. If Milagro is run in parallel with a replicated spatial domain, the calculation of the emission cdf will be replicated on each processor for the entire spatial mesh. Thus, fully replicated multigroup Milagro effectively has more serial overhead than fully replicated gray Milagro and will not scale as well. Run-times with increasing number of groups for a 2-processor, domain-replicated calculation are shown in Fig. 3. The calculation that produced the plots in Fig. 3 had a relatively small amount of transport work each cycle. Therefore, the small increase in transport work for more groups is due to the extra sampling, and the larger increase in overall time is due to the extra cdf calculations in the source.

In the case of domain decomposition for both gray and multigroup Milagro, the parallel efficiency and scalability depends on the problem.

3.4.2. *Statistical Noise.*

In a gray IMC calculation, the frequency variable is analytically integrated out. Whenever stochastic parts of a Monte Carlo calculation can be replaced easily with analytic solutions, statistical noise decreases and overall efficiency increases. Therefore, the user should use a gray frequency treatment if it is appropriate for the problem.

Running multigroup IMC adds another independent variable—another degree of freedom. An increase in phase space or a refinement in discretized phase space requires more particles to adequately sample that phase space. It is up to the user to request more IMC particles to adequately sample the added frequency variable.

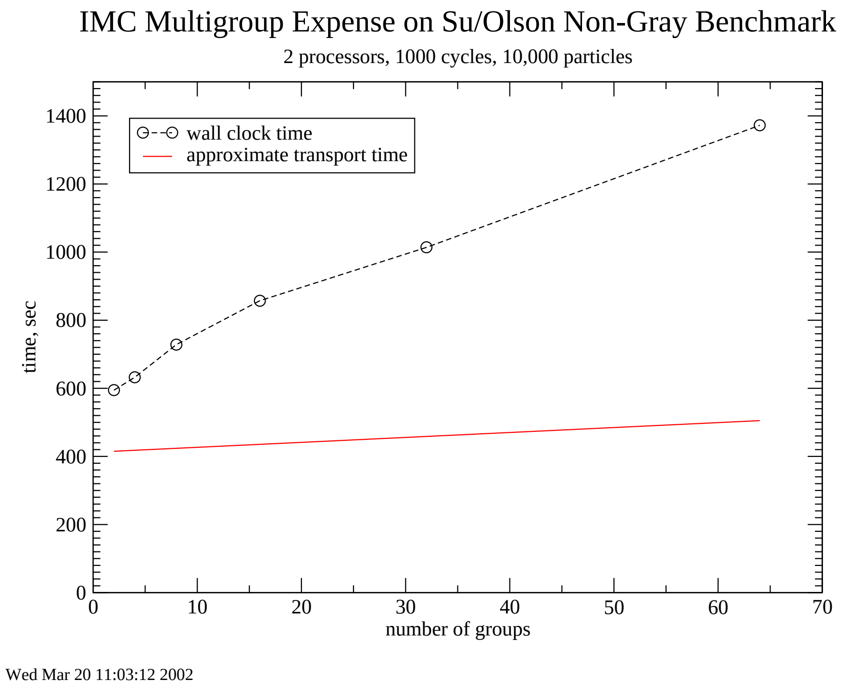

FIG. 3: Parallel efficiency for domain-replicated, multigroup IMC.

4. **Software Design**

We did not want to throw away the gray capability when we introduced the multigroup capability to Milagro. We discovered that in order to keep both capabilities and yet have a safe, efficient code, we needed something that C++ could not give us: partial specialization of member functions (something most people feel C++ should allow). Therefore, we devised a new approach that allows partial specialization of member functions and is standard-compliant.

Contrast our approach with the "old" coding practice of littering the code with "if" checks: if gray, do this, or if multigroup, do something else. "if" checks at such a deep level are dangerous and inefficient. For example, it would be easy for the gray coding to get called accidentally during a multigroup execution. Unnecessary "if" checks do not belong in computationally intensive components, and they provide a weak compartmentalization between disparate capabilities. Weak compartmentalization compromises the ability to test components.

We describe how we added multigroup without losing the gray capability.

4.1. **Template Parameters**

One familiar tactic we have used in Milagro is that we have written the physics classes in a generic way so that they work with any mesh-type—as long as the mesh-type provides the required services (e.g., distance-to-boundary, volume, and next_cell). This sort of code development allows for code-reuse, more efficient code execution, and easier testing.

For upgrading to multigroup, we introduced a new template parameter called "Frequency Type," which can be either "Gray_Frequency" or "Multigroup_Frequency" exclusively. Let us consider as an example the Planckian source energy calculations discussed above. In those calculations, we introduced a function called "get_integrated_norm_Planck," which, besides having a catchy name, returns hard unity for the Gray_Frequency case and $\int_{\nu_0}^{\nu_G} b_\nu(T)\ d\nu$ for the Multigroup_Frequency case. Milagro is configured and compiled such that only one of the two cases is ever true.

4.2. **Explicit Partial Specializations of Templated Member Functions**

As described above, we template physics classes on the `Frequency Type`. There are two primary benefits to this model: (1) there are no conditionals littered throughout the code switching between gray and multigroup physics; (2) the code is highly optimized for both gray and multigroup because logical execution paths are decided at compile time, not run-time. At this point a fair question is, "Why use templates instead of inheritance?" The answer is that the physics code components already support multiple parallel schemes through inheritance. Adding another level for frequency discretization would have made the inheritance hierarchies multi-level and confusing. Furthermore, the physics components are already templated on the spatial discretization (`Mesh Type`); thus, adding the frequency discretization as a template parameter gives the code an attractive consistency and symmetry.

The negative associated with this choice is that there is no real *concept* associated with the `Frequency Type` (`FT`). For example, the `Mesh Type` concept defines a certain set of services that **all** meshes must provide, distance-to-boundary and so forth. There is no corresponding set of services that all `Frequency Types` must define. In fact, the `Gray_Frequency` and `Multigroup_Frequency`, define orthogonal sets of operations. This leads to a template *specialization* model. In the specialization model, an implementation of a template class is explicitly provided for each (but not necessarily all) template arguments. When a template specialization depends only on one of multiple template arguments, the result is called a partial template specialization.

The template specialization model works very well for classes that are fundamentally different for each specialization. For example, the specializations of `Opacity<FT>` with `Gray_Frequency` and `Multigroup_Frequency` will be totally different. They will have different member data and different implementations of member functions. In this case, nothing special is required and the C++ class template specialization model works well.

However, there are cases where the majority of the class will be the same for either template argument. In this case it is a waste of effort to make two versions of the class, the majority of which is the same, for the small percentage that is different. More than wasted effort, replicating code that performs identical operations for identical reasons is inherently dangerous from a maintenance standpoint. One could design "helper" classes that perform the common operations. The danger with this choice is that there is no simple mechanism to prevent users from grabbing these helper classes and utilizing them in undefined and undesirable ways[1]. Unfortunately, C++ does not allow partial specialization of member functions. However, we can use overloading combined with partial ordering to devise a solution [15].

[1]One could be protected by using a combination of friendship and private static functions. This still leads to a somewhat cumbersome solution.

Consider the `Flat_Mat_State_Builder` class from the Draco IMC package.

```
template<class MT, class FT>
class Flat_Mat_State_Builder
    : public Mat_State_Builder<MT,FT>
{
  public:
    // Useful typedefs.
    typedef rtt_dsxx::SP<MT>                                SP_Mesh;
    typedef rtt_dsxx::SP<Mat_State<MT> >                    SP_Mat_State;
    typedef rtt_dsxx::SP<FT>                                SP_Frequency;
    typedef rtt_dsxx::SP<Opacity<MT,FT> >                   SP_Opacity;
    typedef std::vector<double>                             sf_double;
    typedef std::vector<sf_double>                          vf_double;
    typedef rtt_imc::global::Type_Switch<Gray_Frequency>       Switch_Gray;
    typedef rtt_imc::global::Type_Switch<Multigroup_Frequency> Switch_MG;
    typedef rtt_dsxx::SP<Flat_Data_Container>               SP_Flat_Data;
    typedef rtt_dsxx::SP<Opacity<MT,Gray_Frequency> >       SP_Gray_Opacity;
    typedef rtt_dsxx::SP<Opacity<MT,Multigroup_Frequency> > SP_MG_Opacity;
    typedef rtt_dsxx::SP<Gray_Frequency>                    SP_Gray;
    typedef rtt_dsxx::SP<Multigroup_Frequency>              SP_MG;

  private:
    // Flat, cell-centered data fields received from the interface.
    SP_Flat_Data flat_data;

    // Densities in g/cc.
    sf_double    density;

    // Material temperatures in keV.
    sf_double    temperature;

    // Fleck and Cummings implicitness factor.
    double       implicitness;

    // Timestep in shakes.
    double       delta_t;

  private:
    // >>> PARTIAL SPECIALIZATIONS ON FREQUENCY TYPE

    // Build a Gray_Frequency.
    template<class Stop_Explicit_Instantiation>
    rtt_dsxx::SP<Gray_Frequency> build_frequency(Switch_Gray);

    // Build a Multigroup_Frequency
    template<class Stop_Explicit_Instantiation>
    rtt_dsxx::SP<Multigroup_Frequency> build_frequency(Switch_MG);

    // Build an Opacity<MT,Gray_Frequency>
    template<class Stop_Explicit_Instantiation>
    SP_Gray_Opacity build_opacity(Switch_Gray, SP_Mesh, SP_Gray, SP_Mat_State);

    // Build an Opacity<MT,Multigroup_Frequency>
    template<class Stop_Explicit_Instantiation>
    SP_MG_Opacity build_opacity(Switch_MG, SP_Mesh, SP_MG, SP_Mat_State);

  public:
    // Constructor.
    template<class IT>
    explicit Flat_Mat_State_Builder(rtt_dsxx::SP<IT>);

    // >>> PUBLIC INTERFACE

    // Build the frequency.
    SP_Frequency build_Frequency();

    // Build the Mat_State.
    SP_Mat_State build_Mat_State(SP_Mesh);

    // Build the Opacity.
    SP_Opacity build_Opacity(SP_Mesh, SP_Frequency, SP_Mat_State);
};
```

In this class, all of the member data is identical whether the `Frequency Type` is gray or multigroup. In fact, the only member functions that require specialization on the `Frequency Type` are

```
SP_Frequency build_Frequency();
SP_Opacity build_Opacity(SP_Mesh, SP_Frequency, SP_Mat_State);
```

As stated above, we cannot implement partial specializations of these functions in the following manner:

```
template<class MT>
SP<Gray_Frequency> build_Frequency() {/*...*/}
template<class MT>
SP<Multigroup_Frequency> build_Frequency() {/*...*/}
```

What we can do is apply overloading with partial ordering to perform the equivalent task[2]. The first step is to provide a method of dispatching the `Frequency Type` as a function argument in a lightweight way. This is accomplished with the following helper class:

```
template<class T>
struct Type_Switch
{
    typedef T Type;
};
```

Using this type as an argument, we can define the following private overloaded functions to build the frequency,

```
typedef Type_Switch<Gray_Frequency>       Switch_Gray;
typedef Type_Switch<Multigroup_Frequency> Switch_MG;

// Build a Gray_Frequency.
```

[2]This is an ANSI-compliant technique, see Sec. 14.5 of the C++ Standard [16]

```
template<class Stop_Explicit_Instantiation>
SP<Gray_Frequency> build_frequency(Switch_Gray);

// Build a Multigroup_Frequency
template<class Stop_Explicit_Instantiation>
SP<Multigroup_Frequency> build_frequency(Switch_MG);
```

We can now implement the `build_Frequency` member function in a type-safe, generic way:

```
template<class MT, class FT>
typename Flat_Mat_State_Builder<MT,FT>::SP_Frequency
Flat_Mat_State_Builder<MT,FT>::build_Frequency()
{
    Check (flat_data);

    // return frequency
    SP_Frequency frequency;

    // build the frequency, specialize on the frequency type
    frequency = build_frequency<Type_Switch<FT>::Type>(Type_Switch<FT>());

    Ensure (frequency);
    return frequency;
}
```

The public interface is unchanged, as it should be. Additionally, the appropriate "specialization" on `Frequency Type` is instantiated and called, resulting in the correct frequency being built. The functions that implement the specializations are private members and, thus, cannot be inappropriately called by client code.

The careful reader may have noticed that there is an additional template argument in these functions. The additional template argument, `Stop_Explicit_Instantiation`, stops explicit instantiations from building overloaded functions on incompatible types. The dispatching technique works because the C++ automatic template instantiation mechanism only instantiates function templates that are used. However, if we write explicit template declarations like the following:

```
template class Flat_Mat_State_Builder<OS_Mesh, Gray_Frequency>;
```

then, by definition, all template members will be instantiated. In this case, the overloaded functions on `Switch_MG` would be needlessly instantiated. This causes two potential problems: (1) functions that cannot be used are instantiated causing code bloat, (2) compiler errors could result if the specialized template argument tries to execute services that are not defined in a given specialization. To prevent an explicit specialization declaration from instantiating the overloaded functions, we add a "dummy" template argument. Thus, we get exactly the desired behavior, which is that only overloaded functions using the specified template argument are instantiated by the compiler. While the dispatch technique for partial instantiation is described in Ref. [15], the use of an extra template parameter to prevent explicit instantiation is believed unique to this project.

5. **Milagro Multigroup Input**

We describe the multigroup modifications to Milagro's input. The basic Milagro input is fully described in the Milagro-2_0_0 release note [6]. The Milagro input is not backward compatible because of the significant changes.

5.1. **Frequency Group Definition**

In the material block of the input for a multigroup calculation (with, e.g., the executable milagro_xyz_mg), the user must now specify the number of groups and the group bounds, in keV:

```
num_groups:   3
group_bounds: 0.0  0.2  3.0  100.0
```

The group boundaries and number of groups only need to be defined when using analytic opacities or when using a combination of IPCRESS and analytic opacities.

5.2. **Analytic Opacities**

For analytic opacities, the user must input the number of opacities, the number of opacity models, and the opacity models (constant or polynomial), as in the following example for a $1/T^3$ absorption opacity and a constant zero scattering opacity. Note that, for analytic opacities, the opacity definition must contain the phrase "model," a key stating whether it is absorption or scattering, and the model number (indexed beginning with 1) *for each group.*

```
num_opacities: 2
num_opacity_models: 2

c opacity models (cm^2/g): constant or polynomial.
c  constant:   specify a
c  polynomial: specify a,b,c,d  in (a + bT^c) * (rho)^d
opacity_model: 1 polynomial  0.0 1.0 -3.0 0.0
opacity_model: 2 constant    0.0

c opacities
opacity: 1 model  absorption  1 1 1
opacity: 2 model  scattering  2 2 2
```

5.3. **Opacities from an IPCRESS File**

Milagro also reads opacity data from an IPCRESS opacity file [7]. Instead of defining an opacity model, the user must specify the IPCRESS opacity filename, whether it is absorption or scattering, whether it is "planck" or "rosseland," and the material ID number in the file. The following example uses the rosseland opacity from material 19000 for the absorption opacity and an analytic zero scattering opacity.

```
num_groups:    3
num_opacities: 2
num_opacity_models: 1
group_bounds: 1.0e-10  0.2  3.0  100.0

c models: constant or polynomial
opacity_model: 1 constant    0.0

c absorption opacities from a Gandolf opacity file
opacity: 1 tcube.ipcress  absorption  rosseland  19000
opacity: 2 model  scattering  1 1 1
```

If analytic and file opacities are used in the same problem, the group boundaries must be input by the user and they must match those in the IPCRESS opacity file, or Milagro will fire an assertion and abort. If only file opacities are used, the group boundaries will be read from the file, i.e., the `num_groups` and `group_bounds` entries are unnecessary.

5.4. **Equation-of-State Definition**

At this time, only analytic expressions of a constant form or polynomial form ($a+bT^c$) are available for the specific heat [Jerks/g/keV], as shown in the following example:

```
num_eos: 1

c analytic_form_of_eos (Jerks/g/keV): (a + bT^c)
c analytic eos models: constant or polynomial
eos: 1 analytic constant 0.1
```

5.5. **Material Definition**

In the material block, the material is defined with a material ID, descriptive name, density [g/cc], initial temperature [keV], absorption opacity ID, scattering opacity ID, and EOS ID. The following is an example:

```
c mat_id, mat_descr, dens(g/cc), init_temp(keV), abs_id, scat_id, eos_id
mat: 1 mat1 1.0 1.0 1 2 1
```

6. **Verification**

Verification is an essential part of introducing new code capabilities. Verification is not as much about code correctness itself as it is about actually *knowing* that the code is correct.

Milagro's multigroup frequency capability is built with rigorously tested elemental components: Planckian integrators, Planckian samplers, cdf sampling, binary searches for frequency groups, opacity builders and accessors, energy calculations, particle packing, particle tracking, input parsing, etc. These components are tested during development and nightly in the automated regression testing [17].

At the executable-code level, the multigroup frequency capability must meet Milagro's normal physics and robustness tests. At this level, such things as restarting, parallelism, and energy conservation are tested. Finally, at the executable-code level, we test Milagro against analytic benchmarks.

6.1. **Multigroup Problems That Are Degenerately Gray**

As with gray Milagro, multigroup Milagro's regression tests double as high-level tests of low-level components or low-level physics [17]. The regression tests fall into the following categories:

- steady-state, infinite, homogeneous medium
- streaming in a void
- problems with only one source-type of particles
  - all surface source particles
  - all volume emission particles
  - all census particles
- restarting
  - same parallel topology
  - replication to decomposition
  - decomposition to replication
- short versions of analytic benchmarks

These problems are run with the new multigroup executable of Milagro, but the group structure and opacity data are set to emulate a gray calculation. Each group has exactly the same opacity. These problems were run with three groups with the following group boundaries: 0.0, 0.2, 3.0, 100.0 keV. These boundaries were chosen to give very nearly a unity value of the integral of the normalized Planckian for a temperature of 1 keV.

The results of these test problems generally are not identical to the gray results because of the extra random numbers required to sample the frequency group. (The exception is the test problem with all census particles.) However, they all give results that are statistically equivalent to the corresponding gray results. We expect this statistical equivalence because the degenerate multigroup case is essentially the gray case with a different set of random numbers.

In addition to the "short versions of analytic benchmarks" that are run nightly in the regression tests, "long" versions are run in the same mode of setting the multigroup parameters to emulate the gray case. One of the "long" problems is the Marshak-2A, which is a less-diffuse variant of the Marshak-2B test problem [18]. Figure 4 shows the Milagro results for the Marshak-2A test problem for both gray and degenerate multigroup cases. As expected, the results agree with each other. The agreement is a zeroth order verification of the multigroup coding.

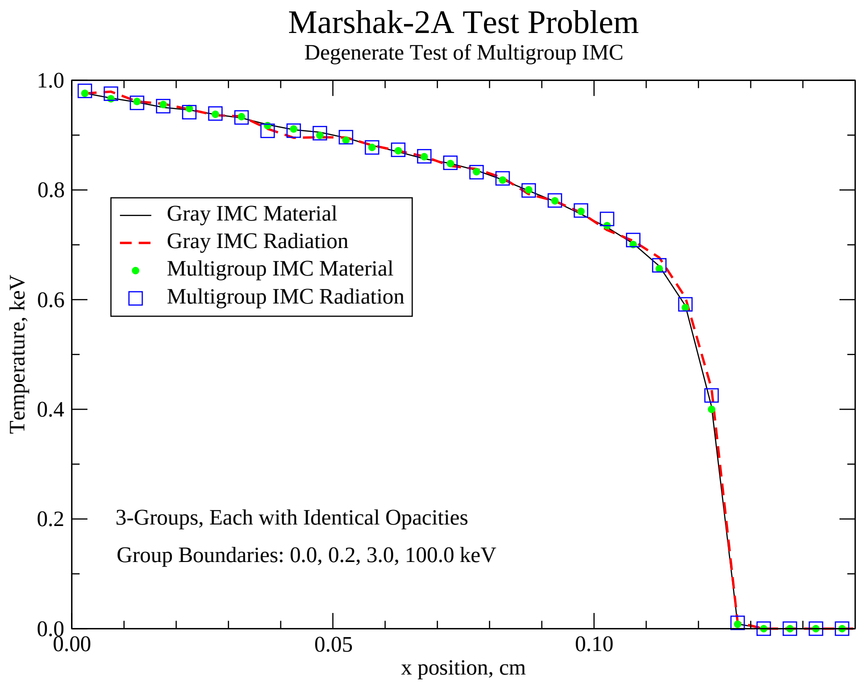

FIG. 4: Zeroth order verification of the multigroup capability in the Milagro IMC code.

6.2. **Su and Olson Non-Grey Benchmarks**

Su and Olson have derived an analytic, multigroup benchmark for non-equilibrium thermal radiative transfer [19]. It is similar to their gray benchmark [20], except that it considers a multigroup picket fence model for its opacity. For a given $d\nu$, the picket fence model has two opacities, of specified fractions, that are independent of temperature and frequency. If a code can model an underlying flat spectral shape, it can run the Su/Olson problem with just 2 groups. Milagro, however, has assumptions of underlying Planckian spectra, so, to run the Su/Olson problem, it requires several groups to make the Planckian change negligible over each group.

The Su/Olson Non-Grey benchmark has two non-grey cases: Case B has one opacity 10 times the other, and Case C has one opacity 100 times the other (Case A is a gray case). The fraction of each opacity in both cases is 50%.

6.2.1. *A Caveat.* We must admit that Milagro cheats on the Su/Olson problems because it analytically updates the material temperature whenever the specific heat is proportional to T^3 [21]. We implemented the analytic update under pressure from colleagues who were analyzing methods designed solely for the linear radiative transfer equations, a set of equations in which the specific heat's T^3 analytically cancels out. No doubt, we are violating the spirit of verification by running special code for certain circumstances. However, from a practical standpoint, we will continue to use the analytic update because it makes our calculations of Su/Olson problems faster. Besides, we hope to extend this analytic temperature update to data tables that have a fitted interpolation shape between points.

6.2.2. *Convergence with Number of Groups.* For the Case B results at $ct = 10$, we considered a frequency range of 0 to 100 keV divided evenly into uniform groups. We show the convergence of Milagro's results to the analytic solution by successive doubling of the number of groups. Figures 5 and 6 show the convergence of the material energy and radiation energy, respectively, with increasing number of groups. Adequate resolution appeared to come at 64 groups. The order of convergence is not of particular interest here, because it merely represents the dynamics for which the in-group Planckian changes become negligible.

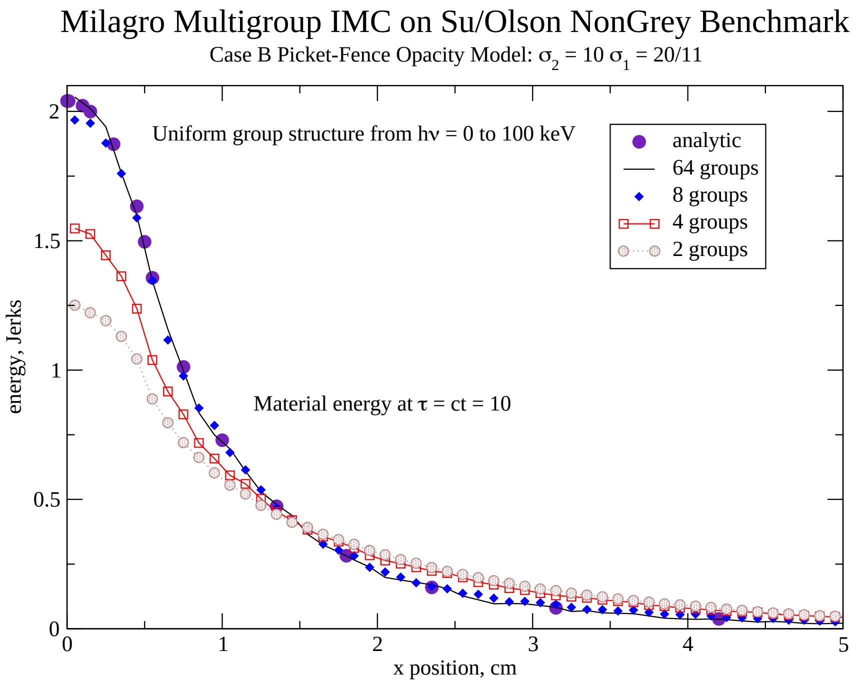

FIG. 5: Convergence of Milagro's material energy with increasing groups (and resolution of the Planckian) for the Su/Olson nongrey benchmark Case B at time $ct = 10$.

Case C of the Su/Olson analytic benchmark has a picket-fence opacity model where one opacity is 100 times the other (as opposed to Case B, where one is only 10 times the other). Case C requires better frequency resolution than Case B, because the greater changes in opacity make the model more sensitive to the in-group Planckian changes. As shown for the material energy in Fig. 7,

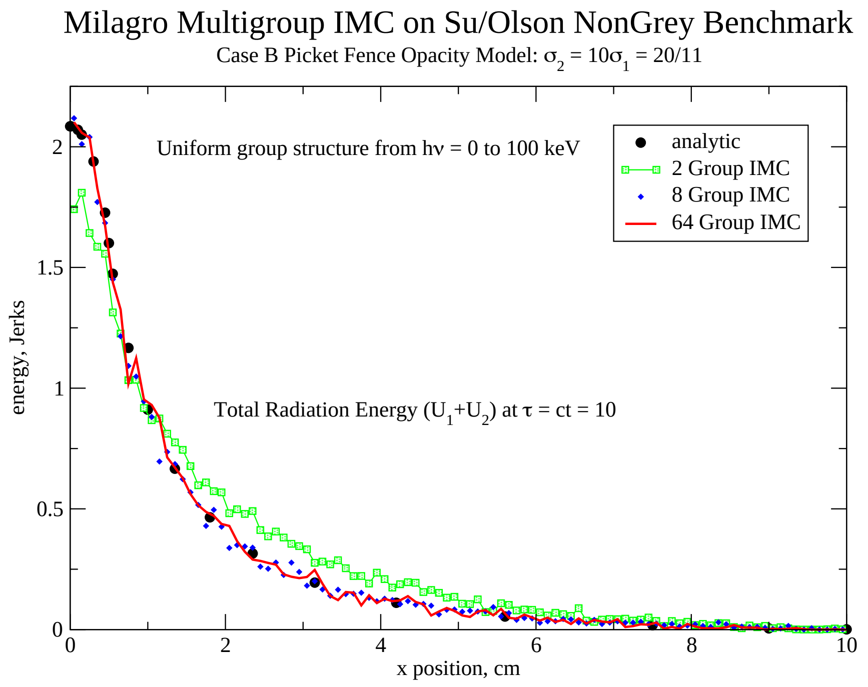

FIG. 6: Convergence of Milagro's radiation energy with increasing groups (and resolution of the Planckian) for the Su/Olson nongrey benchmark Case B at time $ct = 10$.

Case C required at least 128 uniform groups over the range 0 to 102.4. (Note that, for Case C, we changed the upper limit to a value that is more easily halved.)

6.2.3. *Results at all time edits: Cases B and C.* Figures 8 and 9 have the multigroup Milagro IMC results for all time edits of Case B of the Su/Olson nongrey analytic benchmark.

Figures 10 and 11 shows the multigroup Milagro IMC results for Su/Olson's nongrey Case C, which has a picket fence opacity model that is ten times more disparate than Case B.

6.2.4. *Convergence with Δx and Δt at Early Time.* The lack of agreement at early times between the Milagro multigroup IMC results and the Su/Olson analytical solution is obvious in Figures 8 and 10. In general, these errors are due to the numerical method, the algorithm, and modeling specifics. For timesteps that are too large, the Fleck and Cummings method violates the maximum principle theorem and produces nonphysical—but not unstable—results, especially at early time [22, 23]. Also at early time is the extra modeling complication of a step-function source. In the way of algorithms, Milagro's use of time-explicit material properties usually affects the accuracy of its transport solution. However, for the Su/Olson problems, the T_n^3 in the specific heat cancels out in the Fleck factor and results in temperature–independent IMC opacities (effective scattering and effective absorption).

We refined the timestep in the Milagro calculations of the Su/Olson non-grey Case B benchmark until apparent convergence; then we refined the uniform cell width. The plot of the material energy at the first time edit of $ct = 0.1$ is shown in Fig. 12. The relative errors compared to the analytic

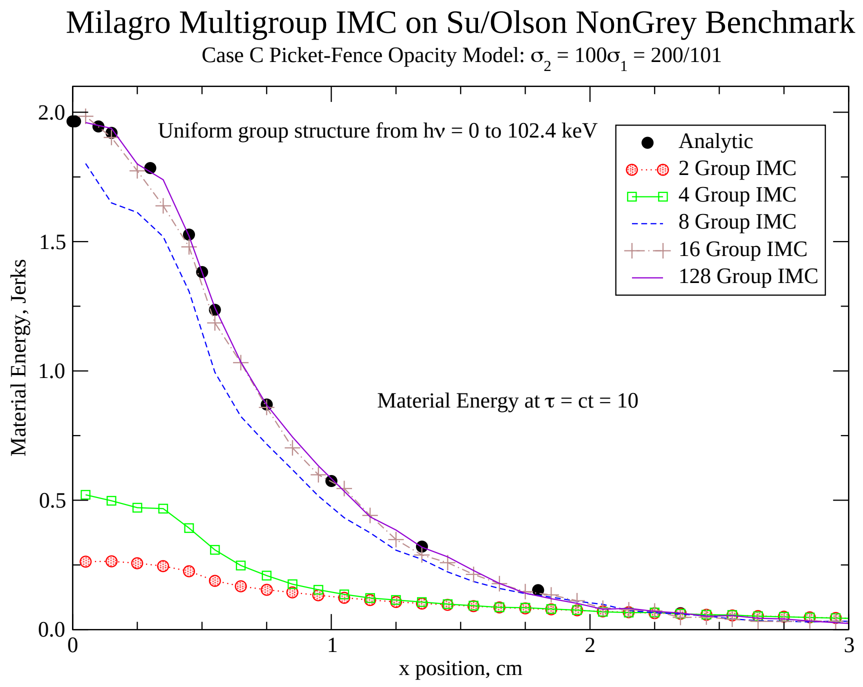

FIG. 7: Convergence of Milagro's material energy with increasing groups (and resolution of the Planckian) for the Su/Olson nongrey benchmark Case C at time $ct = 10$.

solutions are in Fig. 13. We see that the order of both the spatial and temporal error is about 1/2.

Production versions of these calculations would benefit from a dynamic timestep control (initially small, fractional increase, maximum) and zoning that focused refinement around the source's step function at early time.

6.2.5. *Convergence with Number of Particles.* In any traditional linear, unbiased Monte Carlo calculation, the Central Limit Theorem implies that the statistical error (precision, not accuracy) decreases with the number of particles according to an inverse root law,

$$\sigma \sim \frac{1}{\sqrt{N}} \quad . \tag{9}$$

This law is true for any of the IMC tallies made during a timestep.

The Central Limit Theorem holds for the IMC tallies within a given timestep because IMC looks like traditional, linear Monte Carlo within a timestep. However, the overall IMC calculation is nonlinear, so the errors are much more complicated. After a timestep, the statistical results are manipulated by deterministic operations in order to get updated material temperatures. New material properties are calculated or looked-up in data tables with the new temperatures. Thus, the initial conditions of all subsequent timesteps potentially have both statistical and accuracy errors. Propagation of error may occur when statistics are built upon statistics. Bias may occur when deterministic operations are performed on statistical results.

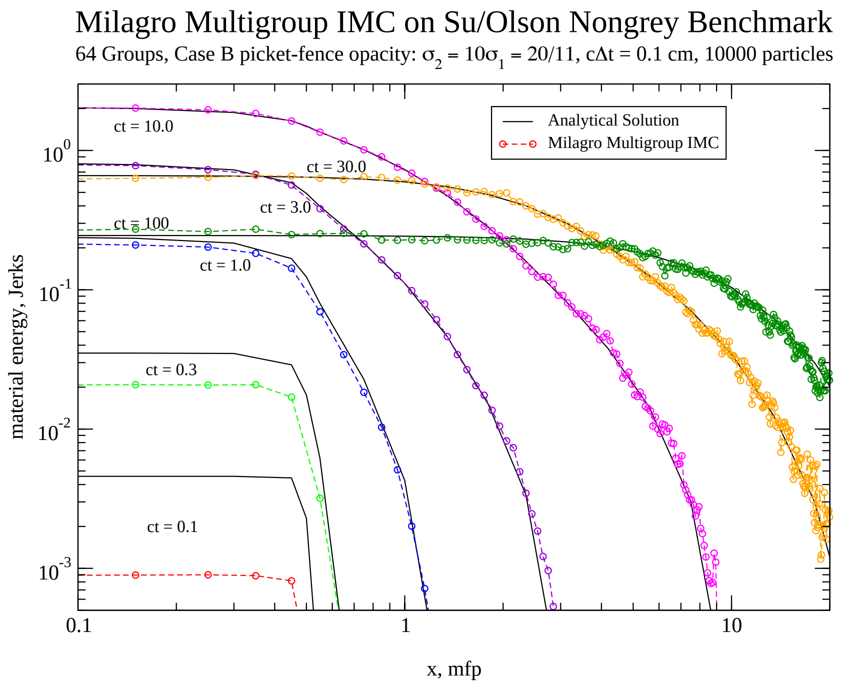

FIG. 8: Milagro multigroup IMC material energies for the Su/Olson nongrey Case B analytic benchmark.

How many particles is enough? The required number of particles in an IMC calculation depends on methods and algorithms. Time-implicit methods and non-analog estimators reduce the number of required particles. Model parameters and the interaction of parameters also affect the number of required particles. Over an IMC timestep, estimates of the energy exchange are constructed by discretizing phase space into bins and accumulating IMC particle tallies during the timestep. Generally speaking, increasing the phase space or, similarly, dividing the same phase space into more bins, requires correspondingly more particles to maintain the same statistical noise per bin. So, for wave propagation problems such as the Su/Olson problems, more particles are required as the energy flows into a larger spatial domain.

Let us consider the IMC errors beyond early time, where the Fleck and Cummings IMC method is well-behaved [22]. Successive IMC timesteps are dependent because of the time-dependent physics; the result of one IMC timestep determines the initial condition for the next IMC timestep. Due to these dependencies, the statistical noise in one timestep may propagate to the next timestep and possibly build up, especially if the noise swamps the dynamic behavior of the physics. However, propagation of error will not be a problem as long as there are enough particles to capture the dynamics each timestep. Whether a number of particles is enough or not depends on the properties of the modeled system. For example, IMC is noisy and inefficient in very diffuse situations.

Dependency implies correlation. A positive correlation between successive IMC timesteps means that, if a result is too large statistically during one timestep, it will likely be too large statistically during the next timestep. Whether correlations affect the propagation of error depends on the number of particles, the size of the timestep, the accuracy errors, and relative time-length scales of

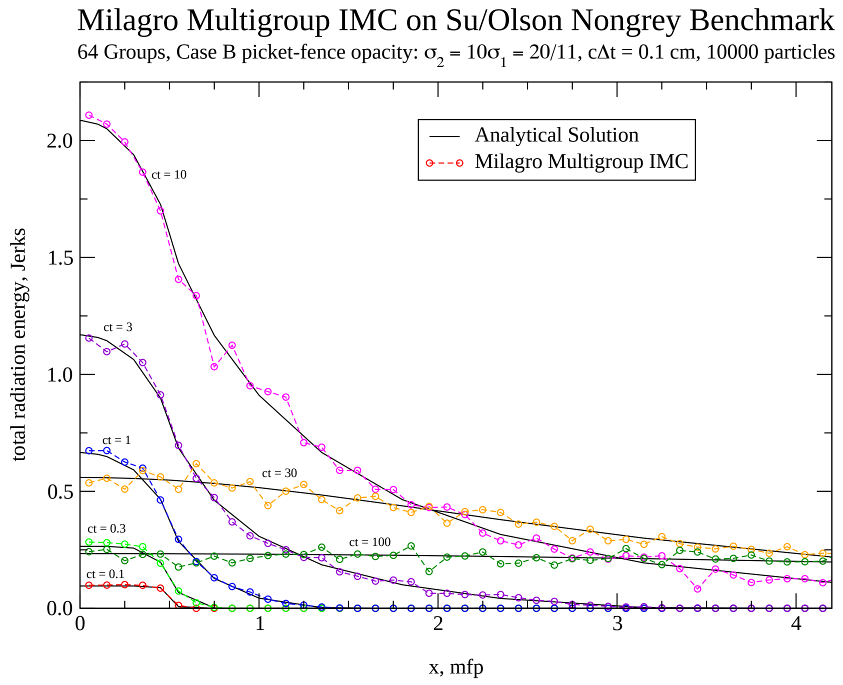

FIG. 9: Milagro multigroup IMC total radiation energies for the Su/Olson nongrey Case B analytic benchmark.

the physics and statistics.

Bias is a term that describes a departure of the expected value of a Monte Carlo result from the true value. One cause of bias in an otherwise unbiased Monte Carlo calculation is deterministic operations on random variables. For example, there is the following Monte Carlo truism: "the ratio of averages is not equivalent to the average of the ratios." In the IMC method, the material temperature update at the end of each cycle involves a deterministic operation on the Monte Carlo estimate of the net energy deposition from the radiation to the material. Having enough particles reduces any such bias to a negligible amount.

Consider Milagro IMC on the Su/Olson nongrey Case C problem. We find that the statistical noise may manifest itself into an apparent negative bias in the material temperature. See Fig. 14 for the difference between IMC calculations using a constant 10^4 particles and a linear ramp-up from 10^4 to 10^6 particles. We believe this apparent bias is due to several things. First, one of the opacities is large enough to produce a diffusive situation, which results in relatively larger statistical noise. Second, the emission, aT^4, when T is particularly noisy, will tend to be relatively larger than statistically expected: given a 10% deviation about unity, 1.0^4 is unity and $(1.0 + 0.1)^4$ is farther away from unity than $(1.0 - 0.1)^4$. Third, the timestep we use, $c\Delta t = 0.1$, is sufficiently small such that, at late time, only about 3% of the radiation is absorbed and re-emitted. The rest stays alive as census particles between timesteps. Therefore, the higher biased radiation energy will not be counteracted by deposition, ΔE, when the material temperature is updated analytically:

$$T_{n+1} = \left(\frac{4\Delta E}{c_{v0}} + T_n^4 \right)^{\frac{1}{4}} \quad . \tag{10}$$

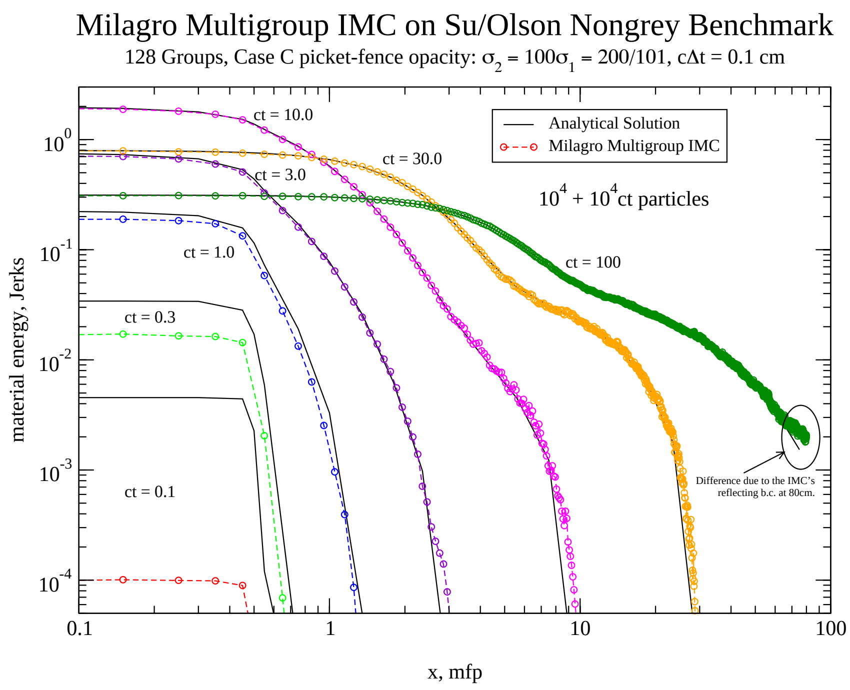

FIG. 10: Milagro multigroup IMC material energies for the Su/Olson nongrey Case C analytic benchmark.

Note that if we had updated the temperature in the normal fashion, where we assume that the specific heat is constant and evaluated at the beginning of the timestep, the bias in the temperature update,

$$T_{n+1} = T_n + \frac{\Delta E}{c_{v_0} T_n^3} \quad , \tag{11}$$

could be much worse than the analytic temperature update. It could produce nonphysically large temperatures because it gives more weight to positive energy depositions (increasing T) than negative energy depositions (decreasing T).

Consider, in Fig. 15, the time-plot of the energies for the calculation with 10^4 particles and the calculation with a linear ramp-up from 10^4 to 10^6 particles. We know that the radiation and material should be equilibrating in this problem, but the calculation with a constant 10^4 particles stagnates and actually deviates from equilibrium. The "snake-like" behavior indicates the presence of positive correlation between timesteps and that, therefore, the actual statistical error is much larger than would be suggested by values from successive timesteps.

7. Package Dependencies

Milagro-3_0_0 depends on the following Draco packages:

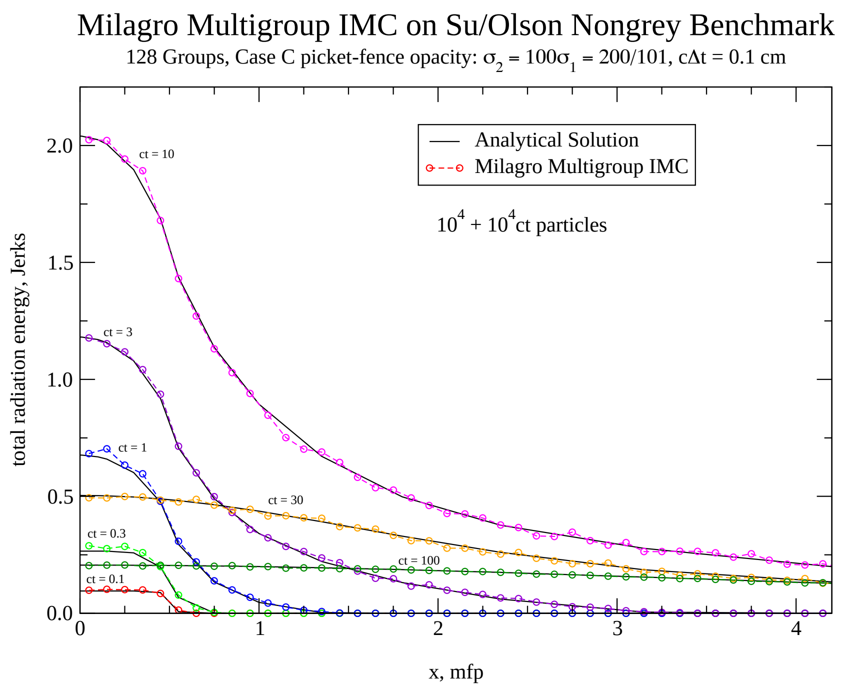

FIG. 11: Milagro multigroup IMC total radiation energies for the Su/Olson nongrey Case C analytic benchmark.

Package	Release
`ds++`	1_6_0
`stdheaders`	1_2_0
`c4`	2_0_0
`traits`	1_4_0
`rng`	1_6_0
`meshReaders`	1_4_0
`cdi`	1_2_0
`viz`	1_3_0
`RTT_Format_Reader`	1_3_0
`cdi_analytic`	1_0_0
`cdi_gandolf`	1_2_0
`mc`	3_0_0
`imc`	3_0_0

8. **Conclusion**

We have released Milagro-3_0_0, the parallel, multi-dimensional, multi-geometry, stand-alone Implicit Monte Carlo (IMC) code for thermal radiative transfer. Milagro-3_0_0 showcases a new multigroup frequency treatment. We have described the assumptions that we took in designing and coding the multigroup capability. We also describe the new C++ techniques we devised that allow Milagro to have both multigroup and gray capabilities without sacrificing efficiency and safety and without introducing redundant code.

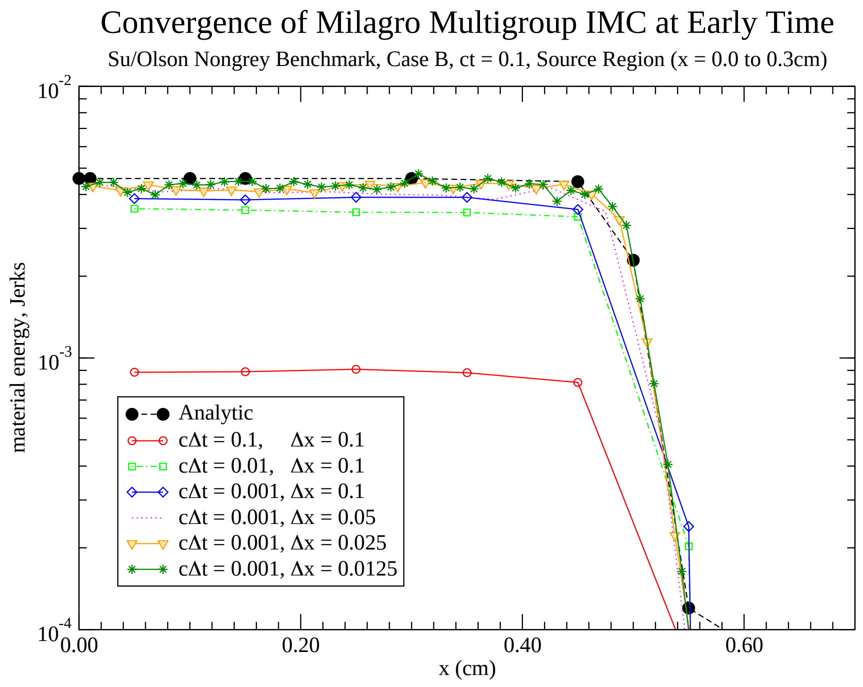

FIG. 12: Early-time convergence of the material energy in Milagro for the Su/Olson non-grey Case B benchmark.

Milagro's new multigroup capability has been verified with a set of degenerate test problems and with the Su/Olson nongrey transport benchmarks. We performed a rudimentary space/time convergence study on the Milagro multigroup results at early time for the Su/Olson Case C problem. We conclude with a discussion on how the number of particles affects error propagation in an IMC calculation.

9. Contacts

Please contact one of us, or our group leader,

- Tom Evans, tme@lanl.gov, 665-3677,
- Todd Urbatsch, tmonster@lanl.gov, 667-3513,
- Mike Buksas, mwbuksas@lanl.gov, 667-7580,
- Gordon Olson, Group Leader, glo@lanl.gov, 667-8105,

to comment on the Milagro IMC code or to request new capabilities.

References

[1] T. M. Evans and T. J. Urbatsch, "MILAGRO: A parallel Implicit Monte Carlo code for 3-d radiative transfer (U)," in *Proceedings of the Nuclear Explosives Code Development Conference*,

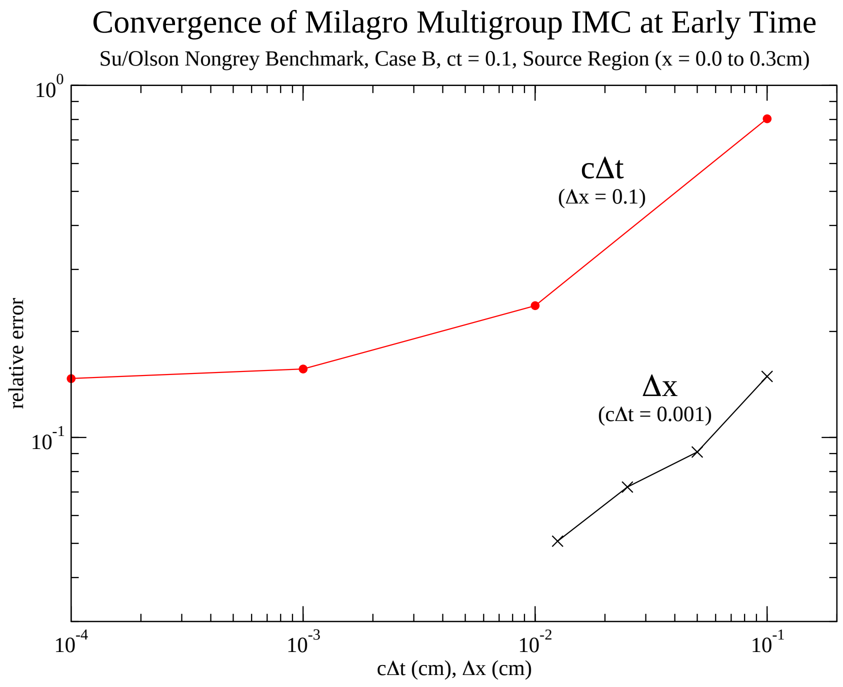

FIG. 13: Early-time relative errors in Milagro's material energy for the Su/Olson non-grey Case B benchmark.

(Las Vegas, NV), Oct. 1998. LA-UR-98–4722.

[2] T. Urbatsch and T. M. Evans, "Release notification: MILAGRO-1_0_0," Research Note XTM:RN(U)99-016, Los Alamos National Laboratory, June 4, 1999. LA-UR-2948.

[3] T. Urbatsch and T. Evans, "Release notification: MILAGRO-1_1_0," Research Note X–6:RN(U)-99-033, Los Alamos National Laboratory, October 26 1999. LA-UR-99-5694.

[4] T. Urbatsch and T. Evans, "Release notification: Milagro–1_2_0," Research Note X-6:RN(U)–99–037, Los Alamos National Laboratory, November 12 1999. LA–UR–99–6087.

[5] T. J. Urbatsch and T. M. Evans, "MILAGRO Implicit Monte Carlo: New capabilities and results (U)," in *Proceedings of the Nuclear Explosives Code Development Conference*, (Oakland, CA), Oct. 2000. LA-UR-00-6118.

[6] T. Urbatsch and T. Evans, "New release, Milagro-2_0_0: Details on development and usage." In development, 2001.

[7] K. G. Thompson, "Gandolf opacity package for draco," Technical Memo CCS-4:01-05(U), Los Alamos National Laboratory, May 2001.

[8] J. A. Fleck, Jr. and J. D. Cummings, "An implicit Monte Carlo scheme for calculating time and frequency dependent nonlinear radiation transport," *Journal of Computational Physics*, vol. 8, pp. 313–342, 1971.

[9] J. A. Fleck, Jr. and E. H. Canfield, "A random walk procedure for improving the computational efficiency of the Implicit Monte Carlo method for nonlinear radiation transport,"

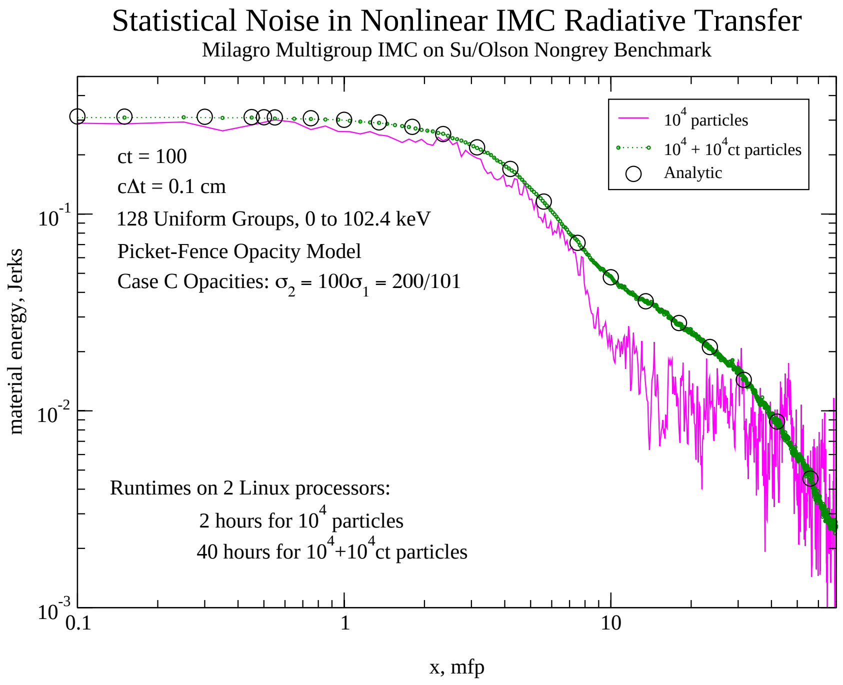

FIG. 14: Late-time statistical noise in Milagro multigroup IMC with too few particles.

Journal of Computational Physics, vol. 54, pp. 508–523, 1984.

[10] K. G. Thompson, "EOSPAC equation of state package for draco," Technical Memo CCS-4:01-17(U), Los Alamos National Laboratory, May 2001.

[11] B. A. Clark, "Computing multigroup radiation integrals using polylogarithm-based methods," *Journal of Computational Physics*, vol. 70, pp. 311–329, June 1987.

[12] T. J. Urbatsch, "A simpler derivation of the definite planck integral (u)," Technical Memo CCS-4-01-28(U), Los Alamos National Laboratory, August 24, 2001 2001.

[13] C. Barnett and E. Canfield, "Sampling a random variable distributed according to Planck's law," Tech. Rep. UCIR-473, Lawrence Livermore National Laboratory, June 1970.

[14] C. J. Everett and E. D. Cashwell, "A third Monte Carlo sampler," Tech. Rep. LA-9721-MS, Los Alamos National Laboratory, March 1983. UC-32.

[15] A. Alexandrescu, *Modern C++ Design*. C++ In-Depth, Boston: Addison-Wesley, Inc., 2001.

[16] ISO/IEC, International Standard, "Programming languages-C++," Tech. Rep. 14882, American National Standard Institute, New York, Sept. 1998.

[17] T. Urbatsch and T. Evans, "Regression testing in Milagro," Research Note XTM-RN(U)-99-018, Los Alamos National Laboratory, June 28 1999. LA-UR-99-3482.

[18] A. G. Petschek, R. E. Williamson, and J. K. Wooten, Jr., "The penetration of radiation with constant driving temperature," Technical Report LAMS–2421, Los Alamos Scientific Laboratory, July 1960.

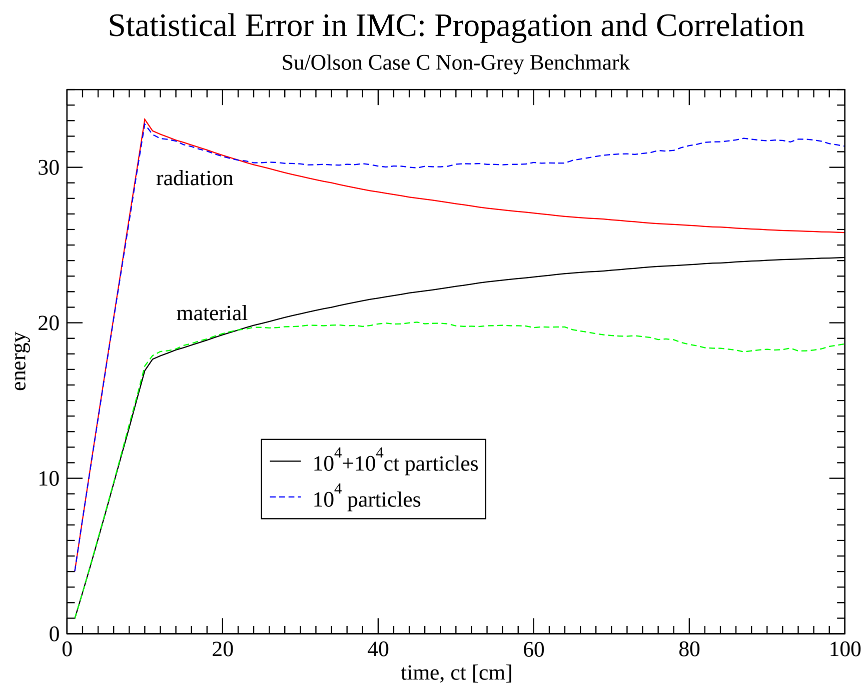

FIG. 15: Apparent bias in an IMC calculation with too few particles.

[19] B. Su and G. L. Olson, “Non-grey benchmark results for two temperature non-equilibrium radiative transfer,” *Journal of Quantitative Spectroscopy & Radiative Transfer*, vol. 62, pp. 279–302, 1999.

[20] B. Su and G. L. Olson, “An analytical benchmark for non-equilibrium radiative transfer in an isotropically scattering medium,” *Annals of Nuclear Energy*, vol. 24, no. 13, pp. 1035–1055, 1997.

[21] T. J. Urbatsch and T. M. Evans, “Analytic temperature updates in milagro for T^3 specific heats,” Technical Memo CCS-4:01-12(U), Los Alamos National Laboratory, March 12 2001. LA-UR-01-1427.

[22] E. W. Larsen and B. Mercier, “Analysis of a Monte Carlo method for nonlinear radiative transfer,” *Journal of Computational Physics*, vol. 71, pp. 50–64, 1987.

[23] W. R. Martin and F. B. Brown, “Error modes in implicit Monte Carlo,” in *Transactions of the American Nuclear Society*, vol. 85, pp. 329–332, American Nuclear Society, 2002.

Distribution:

Ray Juzaitis, ADWP, MS A106
Don Shirk, ADWP, MS F652
David Harris, X–2, MS B220
Brad Beck, X–2, MS B220
Richard Bowers, X–2, MS B220
Bob Chrien, X–2, MS B220
Mike Gittings, X-2/AFF, MS B220
Joyce Guzik, X–2, MS B220
Glenn Magelssen, X–2, MS B220
Jas Mercer-Smith, X–2, MS B220
Charles Nakhleh, X–2, MS B220
Dave Nystrom, X–2, MS B220
Kim Simmons, X–2, MS B220
John Vandenkieboom, X–2, MS B220
Robert Weaver, X–2, MS B220
Dan Weeks, X–2, MS B220
Bernhard Wilde, X–2, MS B220
Blake Wood, X–2, MS B220
Tony Scannapieco, X–3, MS D413
Marv Alme, X–3, MS D413
Tom Gorman, X–3, MS D413
Bob Webster, X–3, MS D413
Don Burton, X–4, MS F664
Alexandra Heath, X–5, MS F663
Forrest Brown, X–5, MS F663
Art Forster, X–5, MS F663
Bill Feiereisen, CCS–DO, MS B297
Stephen Lee, CCS–DO, MS B297
Jim Kamm, CCS–2, MS D413
Bill Rider, CCS–2, MS D413
Gordon Olson, CCS–4, MS D409
Grady Hughes, CCS–4, MS D409
Todd Adams, CCS–4, MS D409
Ray Alcouffe, CCS–4, MS D409
Archuleta Denise G. CCS–4, MS D409
Randy Baker, CCS–4, MS D409
Paul Batcho, CCS–4, MS D409
Michael Buksas, CCS–4, MS D409
David Carrington, CCS–4, MS D409
Brad Clark, CCS–4, MS D409
Jon Dahl, CCS–4, MS D409
Tom Evans, CCS–4, MS D409
Mark Gray, CCS–4, MS D409
Aimee Hungerford, CCS–4, MS D409
Henry Lichtenstein, CCS–4, MS D409
John McGhee, CCS–4, MS D409
Jim Morel, CCS–4, MS D409
Shawn Pautz, CCS–4, MS D409
Kelly Thompson, CCS–4, MS D409
Scott Turner, CCS–4, MS D409
Todd Urbatsch, CCS–4, MS D409
Todd Wareing, CCS–4, MS D409
Jim Warsa, CCS–4, MS D409
CCS–4 Files, MS D409
CCS Files, MS B297

TJU:tju